

A DECISION PROCEDURE FOR HERBRAND FORMULAE WITHOUT SKOLEMIZATION

TIMM LAMPERT

Humboldt University Berlin, Unter den Linden 6, D-10099 Berlin
e-mail address: lampertt@staff.hu-berlin.de

ABSTRACT. This paper describes a decision procedure for disjunctions of conjunctions of anti-prenex normal forms of pure first-order logic (FOLDNFs) that do not contain \vee within the scope of quantifiers. The disjuncts of these FOLDNFs are equivalent to prenex normal forms whose quantifier-free parts are conjunctions of atomic and negated atomic formulae (= Herbrand formulae). In contrast to the usual algorithms for Herbrand formulae, neither skolemization nor unification algorithms with function symbols are applied. Instead, a procedure is described that rests on nothing but equivalence transformations within pure first-order logic (FOL). This procedure involves the application of a calculus for negative normal forms (the NNF-calculus) with $A \dashv\vdash A \wedge A$ (= $\wedge I$) as the sole rule that increases the complexity of given FOLDNFs. The described algorithm illustrates how, in the case of Herbrand formulae, decision problems can be solved through a systematic search for proofs that reduce the number of applications of the rule $\wedge I$ to a minimum in the NNF-calculus. In the case of Herbrand formulae, it is even possible to entirely abstain from applying $\wedge I$. Finally, it is shown how the described procedure can be used within an optimized general search for proofs of contradiction and what kind of questions arise for a $\wedge I$ -minimal proof strategy in the case of a general search for proofs of contradiction.

Keywords: first-order logic; decision procedure; Herbrand formulae; proof search; anti-prenex normal forms.

1. INTRODUCTION

Definition 1.1. A *Herbrand formula* is a formula in prenex normal form whose matrix is a conjunction of atomic or negated atomic formulae.

[Boerger et al.], section 8.2.2, describe a decision procedure for the contradictoriness of Herbrand formulae that reduces the decision problem for such formulae to the unification problem, which is decidable; cf. [Boerger et al.], Theorem 8.2.1. This reduction presumes skolemization. Consequently, the unification algorithm concerns not only bound variables but also function symbols. The reduction of the decision procedure to the unification problem is based on the condition that no variable occurs in two literals after skolemization and subsequent equivalence transformations. This condition is satisfiable in the case of Herbrand formulae. However, it is very specific. For this reason, it is difficult to transfer strategies related to such a decision procedure to other types of first-order formulae.

As an alternative to this approach, I will describe in the following how the contradictoriness of Herbrand formulae can be decided without skolemization. I will refer to nothing

but equivalence transformations of negative normal forms (NNFs) of pure first-order logic (FOL), without names, free variables, function symbols or identity. The decision problem for contradictoriness of Herbrand formulae will also be reduced to a problem of unifying literals. However, unifying literals through a logically valid proof procedure within FOL amounts to answering the question of to what extent universally quantified variables can be replaced with existentially quantified variables in NNFs. There is no need to refer to function symbols and, thus, to go beyond pure FOL for deciding the contradictoriness of Herbrand formulae. Consequently, there is also no need to refer to the axiom of choice for justifying skolemization; cf. the proof of Lemma 2.1.9 in [Boerger et al.], p. 26. Instead, the decision procedure presented in this paper rests on considerations concerning pure FOL proofs and nothing else.

I begin from FOL formulae expressed as NNFs.

Definition 1.2. *Negative normal forms* (NNFs) are formulae that contain only \neg , \wedge and \vee as sentential connectives, with \neg occurring only directly in front of atomic expressions.

Any formula of FOL can be converted into an NNF through equivalence transformations.¹

Herbrand formulae are NNFs in prenex normal form. There is no need to presume prenex normal forms as long as \vee does not occur in the NNFs of interest.

Definition 1.3. \wedge -NNFs are NNFs without \vee .

Any disjunction is contradictory iff each disjunct is contradictory. Therefore, a disjunction is decidable if each disjunct is decidable. Consequently, the greatest profit in applying a decision procedure for \wedge -NNFs arises from driving \vee outward as far as possible by generating FOLDNFs, i.e., disjunctions of conjunctions of anti-prenex normal forms, which are expressed as NNFs. For a convenient algorithm for converting any FOL formula into a FOLDNF of minimal length 1, cf. [Lampert(2017a)]. The procedure described within this paper enables a decision regarding whether a FOLDNF A is contradictory if each disjunct of A is a \wedge -NNF.

The decision problem for Herbrand formulae or \wedge -NNFs can be reduced to a decision problem for \wedge -NNFs ψ with only two literals.²

Definition 1.4. The *subformulae* ψ of a \wedge -NNF ϕ are formulae that are generated by deleting all but two literals, L1 and L2, in ϕ and retaining only those quantifiers that bind variables in L1 and L2.

Theorem 1.5. *A \wedge -NNF ϕ is contradictory iff at least one of its subformulae ψ is contradictory.*

Proof. Any proof of the contradictoriness of a \wedge -NNF ϕ in the tree calculus (tableaux) can be reduced to a proof of the contradictoriness of ψ ; all that must be done is to eliminate quantifiers in the same manner as in the proof of the contradictoriness of ψ . \square

¹Cf., e.g., [Nonnengart & Weidenbach], section 3.2.

² I refer to *two occurrences* of literals and not to an arbitrary number of occurrences of *two types* of literals.

I will describe in sections 6 to 11 how one can decide whether a subformula ψ is contradictory by considering whether the pair $\{L1, L2\}$ is unifiable. In the preceding sections 2 to 5, I will establish general presuppositions for proofs of contradiction for NNFs. In sections 12 to 13, I will explain how to make use of the decision regarding the possibility of unifying $\{L1, L2\}$ to optimize the search for general proofs of contradiction. Finally, in section 14, I will consider what questions arise when generalizing the strategy of the described decision procedure.

2. UNIFICATION

Proofs of contradiction depend on the unification of pairs of connected literals. This section explains what that means.

I will use subscripted x variables (' x variables' for short) to represent variables in NNFs that are bound by universal quantifiers and subscripted y variables (' y variables' for short) to represent those bound by existential quantifiers. Furthermore, I assume that the NNFs are maximally subscripted.

Definition 2.1. *Maximally subscripted* NNFs are NNFs in which any two quantifiers bind different variables.

Any NNF can be transformed into a maximally subscripted NNF with universally quantified x variables and existentially quantified y variables. To do so, the variables must be renamed by means of SUB1 and SUB2; cf. table 1.

$$\begin{array}{ll} \exists\mu A(\mu) \dashv\vdash \exists\nu A(\nu/\mu) & \text{Restriction: } \nu \text{ does not occur in } A(\mu) \quad \text{SUB1} \\ \forall\mu A(\mu) \dashv\vdash \forall\nu A(\nu/\mu) & \text{Restriction: } \nu \text{ does not occur in } A(\mu) \quad \text{SUB2} \end{array}$$

Table 1: Laws of Substitution

The question of unifiability is whether x variables can be replaced with y variables in a maximally subscripted NNF such that identical y variables occupy identical positions³ in connected literals. A pair of connected literals of a maximally subscripted NNF is defined as follows:

Definition 2.2. Two literals L1 and L2 of a maximally subscripted NNF with universally quantified x variables and existentially quantified y variables constitute a *pair of connected literals* iff

- (1) L1 is not negated, whereas L2 is negated.
- (2) L1 and L2 contain the same predicate letter φ .
- (3) The same number of positions succeed φ .
- (4) If position n in L1 is occupied by y_i , then position n in L2 is not occupied by y_j for $i \neq j$.

Manifestly, these conditions are necessary for L1 and L2 to constitute a unifiable pair of literals that might contribute to a proof of contradiction. A maximally subscripted NNF that does not contain any pair of connected literals cannot be contradictory. Thus, the decision problem for subformulae ψ is reduced to the corresponding problem for those

³Identical positions in two literals L1 and L2' means 'the same position in the propositional functions involved'. Thus, I ignore the negation sign that might occur in a literal. In $\{Fx_1y_2, \neg Fy_1y_2\}$, for example, x_1 and y_1 are in identical positions, as are the two occurrences of y_2 .

subformulae ψ that contain a pair of connected literals L1 and L2. In sections 6 to 11, I will consider only subformulae ψ of this kind.

I adopt the following narrow understanding of unification.

Definition 2.3. The *unification* of a pair of connected literals in an NNF is the logically valid procedure of replacing x variables such that identical positions in a pair of connected literals are occupied by identical y variables. A pair of connected literals is *unified* if all of its positions are occupied by identical y variables. A pair of connected literals is *unifiable* if it is possible to unify it through unification.

It is presumed that during unification, *all* universally quantified x variables are ultimately eliminated, even in cases such as $\{Fx, \neg Fx\}$.

Unification in the sense of Definition 2.3 is independent of skolemization. In the context of maximally subscripted NNFs, nothing matters for unification but the replacement of x variables with y variables. Unification must obey logical laws. The problem of unification concerns the replacement of x variables with y variables by means of universal quantifier elimination within a correct and complete calculus for NNFs. Before explaining a procedure for solving this problem, I will first establish the calculus that I will adopt for proofs of contradiction for NNFs. I refer to a calculus that applies not only to \wedge -NNFs but to NNFs in general because it is desired that it should be possible to extend the considered proof strategies to NNFs in general.

3. NNF-CALCULUS

This section establishes a correct and complete calculus for NNFs that I call the ‘NNF-calculus’. As in the tree calculus, proofs in the NNF-calculus are proofs of contradiction: what is to be proven is not that a formula is a theorem but that a formula is a contradiction. Unlike in the tree calculus, however, neither free variables nor names are used. Instead, any proof operates within the realm of *pure* FOL. Also, in contrast to the tree calculus, the NNFs are not decomposed within proofs in the NNF-calculus. Before universal quantifiers are eliminated for unification, equivalence transformations are applied. Universal quantifier elimination is the last step in a proof of contradiction for the sake of unification in the case that a formula is contradictory; cf. section 4.

The rule for universal quantifier elimination is as follows:

$$\exists\mu \dots \forall\nu A(\mu, \nu) \vdash \exists\mu A(\mu, \nu/\mu) \quad \forall E$$

Table 2: Universal Quantifier Elimination

If $\forall\nu$ is within the scope of $\exists\mu$, then the universal quantifier $\forall\nu$ is eliminated, and all occurrences of ν in the scope of $\forall\nu$ are replaced with μ . It is also allowed to replace ν with μ when $\exists\mu$ is a *new* existential quantifier preceding the resulting formula. I subsume this case under $\forall E$. I arbitrarily choose the variable y_0 as a new y variable and require that y_0 does not occur in the expression to the left of \vdash in $\forall E$.

In addition to $\forall E$, the following rule is crucial for proofs of contradiction in the NNF-calculus:

$$A \dashv\vdash A \wedge A \quad \wedge I$$

Table 3: Conjunct Multiplication

In proofs of contradiction in the NNF-calculus, $\wedge I$ is used to multiply universally quantified expressions to the effect that x variables of different conjuncts are replaced with different y variables following the generation of suitable maximally subscripted prenex normal forms; cf. Example 4.1, p. 7. By virtue of this multiplication of universally quantified expressions, one can avoid the need for a rule for existential quantifier elimination and the need to decompose the NNFs. The ability to solve decision problems involving NNFs within the NNF-calculus depends on the extent to which it is possible to define criteria that can be used to decide which universally quantified expressions must be multiplied how often in order to prove that a formula is a contradiction through the ultimate application of $\forall E$. As we will see, the establishment of such criteria depends on equivalence transformations that include the application of $\wedge I$.

Proving a contradiction by means of unification within the NNF-calculus results in a DNF matrix in which each disjunct contains a literal and its negation.

Definition 3.1. The *DNF matrix* of a formula ϕ is the scope that one obtains if ϕ is transformed into a prenex normal form and the resulting scope of that prenex normal form is then transformed into a disjunctive normal form (DNF).

Definition 3.2. An *explicit* contradiction is a formula without universal quantifiers and with a DNF matrix in which any disjunct contains a literal and its negation.

In proofs of contradiction in the NNF-calculus, the aim is to derive explicit contradictions by unifying connected literals through the application of $\forall E$ subsequent to equivalence transformations that include the application of $\wedge I$.

In addition to $\forall E$ and $\wedge I$, the NNF-calculus comprises well-known rules for generating diverse normal forms, such as (i) definitions of sentential connectives and quantifiers as well as De Morgan's laws for generating NNFs, (ii) distributive laws for generating disjunctive and conjunctive normal forms (DNFs and CNFs), and (iii) PN laws (cf. table 5) for generating prenex and anti-prenex normal forms. Finally, we presume the laws of substitution for renaming variables (cf. table 1), associative and commutative laws for \wedge and \vee , and commutative laws for sequences of universal or existential quantifiers. With the exception of $\forall E$, all rules are well-known logical equivalence rules.

Theorem 3.3. *The NNF-calculus is correct and complete.*

Proof. The correctness of the NNF-calculus follows from the fact that all of its rules are well-known correct derivation rules. Its completeness can be proven by showing that any proof within the correct and complete tree calculus can be transformed into a proof in the NNF-calculus. Each quantified expression that is decomposed n times in a proof within the tree calculus is multiplied by applying $\wedge I$ $n - 1$ times in the corresponding proof within the NNF-calculus. Subsequently, one generates a prenex normal form in which any quantifier Y that is eliminated later than a quantifier X on the same proof path in the tree calculus is in the scope of X . Finally, $\forall E$ is applied such that the same literals are unified. The resulting DNF matrix is contradictory iff all proof paths within the tree calculus are closed. \square

In section 5, a further rule, $\exists M$ (cf. table 4), will be added to the NNF-calculus, but only to avoid applications of $\wedge I$ in proofs of subformulae ψ . We will prove that any number of applications of $\exists M$ can always be replaced with one application of $\wedge I$ when proving that a subformulae ψ is contradictory; cf. Theorem 8.4.

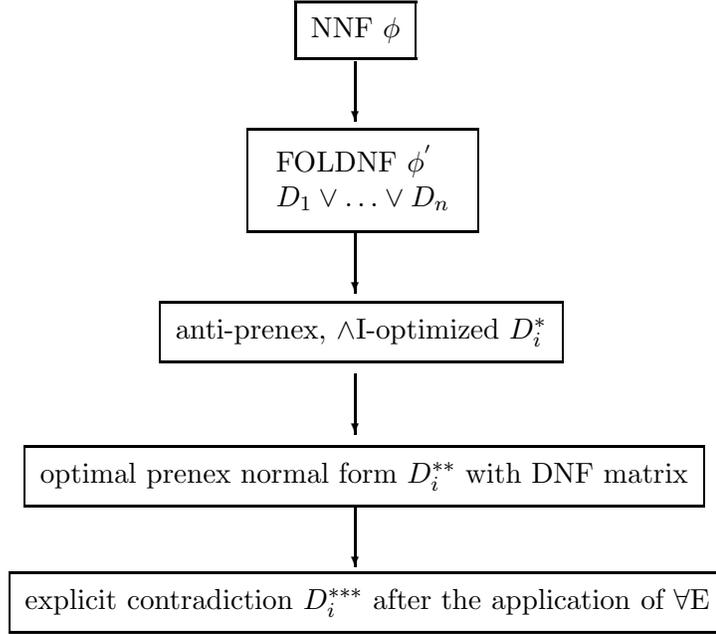


Figure 1: Steps of proof in the NNF-calculus

4. PROOFS IN THE NNF-CALCULUS

This section provides a brief general overview of the essential steps of proofs of contradiction in the NNF-calculus. The general strategy for such proofs also underlies the specific decision procedure for subformulae ψ described in the following sections. The specific steps of the proof procedure cannot be explained or justified in detail for the general case of NNFs at this point. In the following sections, the principles of the proof strategy are explained and justified in detail with respect to the special case of subformulae ψ . Terms such as ‘anti-prenex, optimized FOLDNFs’, ‘ $\wedge I$ -optimized proof’, ‘ $\wedge I$ -minimal proof’ and ‘optimal prenex normal forms’ are explained in this section only to the extent that is necessary for a rough understanding.

An NNF ϕ is first transformed into a sat-equivalent, anti-prenex, optimized FOLDNF ϕ' ($= D_1 \vee \dots \vee D_n$). An effective procedure for transforming a FOL formula ϕ into a logically equivalent FOLDNF ϕ' is described in [Lampert(2017a)], section 2. A procedure for optimizing a FOLDNF is introduced in section 12 below. It consists of deleting literals from D_i that are not part of any unifiable pair of literals in D_i . This first step of generating an anti-prenex, optimized FOLDNF drives quantifiers inward to the greatest possible extent by applying PN-laws ($=$ *anti-prenex*, cf. Definition 6.1). This is essential for the desired proof strategy; cf. section 6. By contrast, \vee is driven outward as far as possible, which makes it possible to reduce the decision problem for ϕ to a decision problem concerning single disjuncts of the resulting FOLDNF.

The second step is to compute, for each single disjunct D_i , whether a sat-equivalent, anti-prenex, $\wedge I$ -optimized formula D_i^* can be generated. *$\wedge I$ -optimized* means that $\wedge I$ is applied no more than is necessary in the case that a proof of contradiction can be identified. I refer to the corresponding proofs as *$\wedge I$ -minimal proofs* and to the corresponding proof strategy as the *$\wedge I$ -minimal proof strategy*. This step is the most crucial and problematic one. Section 14 briefly summarizes the questions that arise when computing the applications of

\wedge I in the general case of FOL formulae. This paper avoids answering these questions by restricting all consideration to \wedge -NNFs. It can be shown that in the case of this fragment of FOL, \wedge I must be applied no more than once (or even not at all, if one allows for \exists M) to prove that a \wedge -NNF is contradictory; cf. Theorems 8.4 and 12.3.

As the third step, each D_i^* is transformed into an optimal prenex normal form D_i^{**} with a DNF matrix. A prenex is *optimal* iff the universal quantifier $\forall\nu$ is within the scope of the existential quantifier $\exists\mu$ if ν is to be replaced with μ to derive an explicit contradiction through the application of \forall E; cf. also Definition 10.6, p. 23.

As the last step, \forall E is applied to derive an explicit contradiction D_i^{***} . Each step of the proof is an equivalence transformation with respect to satisfiability. This is also true in the case of this last, non-essential step because the decision-making is independent of this last step and \forall E is applied only if D_i is contradictory.

Figure 1 schematizes proofs in the NNF-calculus in accordance with this proof strategy. Example 4.1 illustrates a \wedge I-minimal proof of contradiction in the NNF-calculus that requires the application of \wedge I.

Example 4.1.

$$\begin{aligned} & \forall x_1 (\forall x_2 \forall x_6 F x_1 x_2 x_6 \vee \forall x_3 \forall x_7 \neg F x_3 x_1 x_7) \wedge \\ & \exists y_1 \forall x_4 \exists y_3 \neg F y_1 x_4 y_3 \wedge \exists y_2 \forall x_5 \exists y_4 F x_5 y_2 y_4 \end{aligned} \quad (4.1)$$

This is a FOLDNF with only one contradictory disjunct D_i that contains \vee . The proof depends on the unification of the two pairs of connected literals $\{F x_1 x_2 x_6, \neg F y_1 x_4 y_3\}$ and $\{F x_5 y_2 y_4, \neg F x_3 x_1 x_7\}$. To unify these pairs, x_1 must be replaced with y_1 in the first pair and with y_2 in the second pair. To do so, \wedge I must be applied once to duplicate the first conjunct of (4.1). We prescribe that the variables of quantifiers occurring within the conjuncts resulting from the application of \wedge I are to be maximally subscripted by increasing the depth of their indices by one. This results in the following formula D_i^* :

$$\begin{aligned} & \forall x_{1_1} (\forall x_{2_1} \forall x_{6_1} F x_{1_1} x_{2_1} x_{6_1} \vee \forall x_{3_1} \forall x_{7_1} \neg F x_{3_1} x_{1_1} x_{7_1}) \wedge \\ & \forall x_{1_2} (\forall x_{2_2} \forall x_{6_2} F x_{1_2} x_{2_2} x_{6_2} \vee \forall x_{3_2} \forall x_{7_2} \neg F x_{3_2} x_{1_2} x_{7_2}) \wedge \\ & \exists y_1 \forall x_4 \exists y_3 \neg F y_1 x_4 y_3 \wedge \exists y_2 \forall x_5 \exists y_4 F x_5 y_2 y_4 \end{aligned} \quad (4.2)$$

I refer to the number of levels of indices of a variable as its depth. Variables of D_i such as x_1 have a depth of one. Each application of \wedge I increases the level of any variables that are bound by quantifiers occurring in the multiplied conjunct by one. For example, x_{1_1} and x_{1_2} each have a depth of 2.

Definition 4.2. A *derivate* of a variable μ of depth 1 is any variable that is identical to μ if one deletes all indices at levels > 1 . A *derivate of a pair of literals* $\{L1, L2\}$ with variables of depth 1 is a pair of literals that is identical to $\{L1, L2\}$ if one deletes all indices at levels > 1 .

In formula (4.2), x_{1_1} and x_{1_2} can be replaced with y_1 and y_2 , respectively, to unify derivates of the original pairs of literals. However, the application of \wedge I and the intended substitutions x_{1_1}/y_1 and x_{1_2}/y_2 cause the proof to now also depend on the unification of the pair of literals $\{F x_{1_2} x_{2_2} x_{6_2}, \neg F x_{3_1} x_{1_1} x_{7_1}\}$. This results in the following set of pairs of

literals:

$$\begin{aligned} & \{\{Fx_1x_2x_6, \neg Fy_1x_4y_3\}, \\ & \{Fx_5y_2y_4, \neg Fx_3x_1x_7\}, \\ & \{Fx_1x_2x_6, \neg Fx_3x_1x_7\}\} \end{aligned} \quad (4.3)$$

The unification of this set of unifiable pairs of literals is sufficient to prove that (4.1) is contradictory. To unify all pairs of literals in (4.3), the x variables must be replaced with y variables in accordance with the following list of substitutions:

$$\begin{aligned} & \{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}, \{x_4, y_4\}, \{x_5, y_5\}, \{x_6, y_6\}, \{x_7, y_7\}, \\ & \{x_8, y_8\}, \{x_9, y_9\}, \{x_{10}, y_{10}\}, \{x_{11}, y_{11}\}, \{x_{12}, y_{12}\}\} \end{aligned} \quad (4.4)$$

A prenex normal form D_i^{**} with an optimal prenex must be generated from (4.2). We specify one possible optimal prenex in the form of an ordered list that contains only the x and y variables:

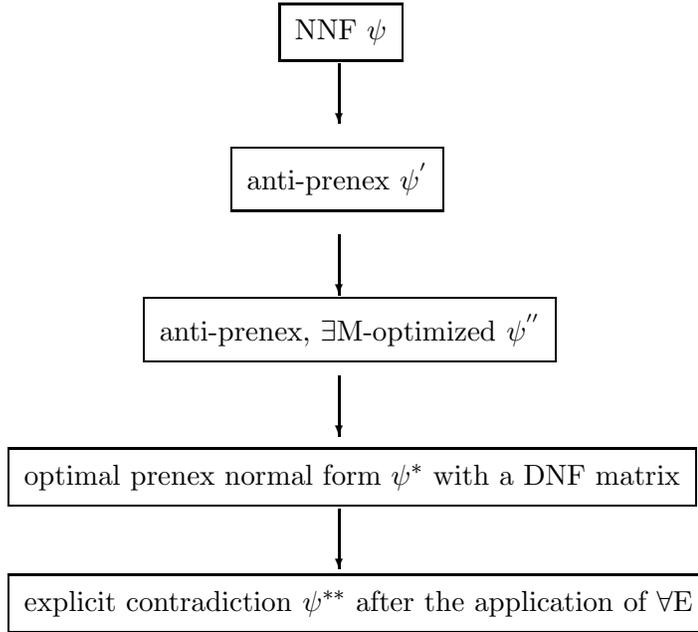
$$\{y_0, y_1, y_2, x_4, y_3, x_5, y_4, x_1, x_2, x_3, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}\} \quad (4.5)$$

Formulae (4.1), (4.2), (4.3), (4.4), and (4.5) serve as a recipe that shows how $\wedge I$ is to be applied to the D_i (4.1) in order to generate the D_i^* (4.2) that is to be transformed into a D_i^{**} with the optimal prenex (4.5) in order to unify the literals in (4.3) by applying $\forall E$ such that an explicit contradiction D_i^{***} is obtained.

The crucial question for the application of the $\wedge I$ -minimal proof strategy is in which cases a decision can be made based on computing the minimal necessary applications of $\wedge I$ within a finite number of steps. The proof strategy ultimately serves the purpose of restricting the number of applications of $\wedge I$ to the minimum necessary. This makes it possible to terminate a proof path when it is recognized that no more applications of $\wedge I$ can be found on that path that may contribute to a $\wedge I$ -minimal proof. Of course, all alternative proof paths that may result in $\wedge I$ -minimal proofs must be generated.⁴ Decidability depends on whether the search tree can be restricted to a finite number of proof paths of finite depth. According to the $\wedge I$ -minimal proof strategy, the computation of the necessary applications of $\wedge I$ must be related to purely syntactic criteria. In this paper, we will study this proof strategy only in the rather simple case of subformulae ψ or, consequently, the case of FOLDNFs in which all disjuncts are \wedge -NNFs; cf. p. 2 above.

I will abstain from considering FOLDNFs in the following sections 6 to 11; instead, I will consider only subformulae ψ . As we will see, one can decide the contradictoriness of subformulae ψ without any application of $\wedge I$. I call the resulting sat-equivalent formulae $\exists M$ -*optimized* formulae. This terminology and the generation of $\exists M$ -optimized formulae are explained in section 7. Figure 2 schematizes proofs for subformulae ψ using $\exists M$ in the NNF-calculus.

⁴Unlike in the tree calculus, different proof paths search for different $\wedge I$ -minimal proofs. Therefore, it is proven that D_i is contradictory as soon as a $\wedge I$ -minimal proof is found on any proof path. By contrast, in the case that all proof paths terminate without the identification of any $\wedge I$ -minimal proof, it is proven that D_i is not contradictory. The question lies in the extent to which such termination can be achieved based on correct criteria for termination; cf. section 14.

Figure 2: Steps of proof for a subformula ψ

5. \wedge I-MINIMAL PROOFS

We are looking for an intelligent algorithm to decide whether explicit contradictions can be derived in the NNF-calculus and, if so, how. The proof strategy that will be exemplified in the following for the case of subformulae ψ consists of a systematic search for \wedge I-minimal proofs by means of equivalence transformations.

Definition 5.1. A proof in the NNF-calculus is *\wedge I-minimal* iff each application of \wedge I in the proof is necessary. An application of \wedge I is *necessary* iff no explicit contradiction could be derived any longer if one of the conjuncts resulting from the application of \wedge I were to be eliminated and all remaining x variables were substituted as before.

Example 5.2. Consider the elimination of the second conjunct in formula (4.2) of Example 4.1, p. 7. Replacing the remaining x variables in accordance with the substitution list given in (4.4) does not result in an explicit contradiction. Therefore, the described proof is \wedge -minimal.

Note that \wedge I-minimal proofs are defined with respect to given proofs of contradiction. It is possible for two different \wedge I-minimal proofs for the same D_i to exist that differ in their numbers of applications of \wedge I. A \wedge I-minimal proof is therefore not necessarily the proof with the smallest number of applications of \wedge I necessary to prove some D_i contradictory.

In a \wedge I-minimal proof search, \wedge I is applied only to unify pairs of literals by means of \forall E. To avoid unnecessary applications of \wedge I, one needs to consider only the *minimal* sets of unifiable pairs that are necessary for proving a contradiction. Likewise, the unification of such a set with substitution list σ_1 must be *minimal*; i.e., no substitution list σ_2 ($\sigma_1 \neq \sigma_2$) that unifies the same minimal set of unifiable pairs of literals may exist such that any member of σ_2 is a part of a member of σ_1 . A \wedge I-minimal proof is also incompatible with an application of \wedge I that, in turn, *induces* another application of \wedge I for the purpose of replacing x variables with y variables in two *similar* conjuncts of D_i^* by means of *similar*

substitutions to unify *similar* pairs of literals; here, conjuncts, substitutions and pairs of literals are considered *similar* iff they are identical after indices at levels > 1 are deleted (cf. the discussion in section 14 on the ‘loop criterion’).

\wedge I-minimal proofs are very elegant; any application of \wedge I is the result of a systematic proof search. I will restrict the following discussion to subformulae ψ . Deciding upon these formulae allows one to identify unifiable pairs of literals. This capability can be utilized for an optimized search for proofs of contradiction; cf. sections 12 and 13. Furthermore, subformulae ψ serve as a simple example to illustrate how the \wedge I-minimal proof strategy can be used to define a decision procedure. This proof strategy is also interesting because the justification of such a decision procedure does not depend on any model theory or semantics; it is sufficient to refer to syntactic considerations concerning the unification of pairs of literals within the NNF-calculus.

Finally, a \wedge I-minimal proof strategy is also interesting because it is efficient, in a certain sense. Proofs of contradiction within the tree calculus become inefficient when quantified expressions are decomposed several times without each decomposition being necessary for the proof. In the NNF-calculus, the same applies to applications of \wedge I for the replacement of x variables with y variables that do not contribute to a proof of contradiction. \wedge I-minimal proofs restrict the elimination of universal quantifiers to the minimum extent necessary for unifying the members of a minimal set of pairs of literals. I will not discuss questions of efficiency any further in this paper because from here on, I will consider only simple subformulae ψ to show that a decision procedure for these formulae is *optimally \wedge I-minimal*.

Definition 5.3. A decision procedure for formulae ϕ is *optimally \wedge I-minimal* if it is possible to decide whether the formulae ϕ are contradictory without applying \wedge I.

In a certain respect, optimally \wedge I-minimal decision procedures correspond to decision procedures with blocking in the tree calculus.⁵ In general, \wedge I is necessary to prove that formulae are contradictory, as illustrated by Example 4.1. In the case of subformulae ψ , however, one can abstain from the application of \wedge I altogether. If ψ is contradictory, then an optimally \wedge I-minimal decision procedure for ψ will find a sat-equivalent formula ψ^* that allows one to unify the pair of literals in ψ^* by applying \forall E without applying \wedge I at all. If ψ is not contradictory, then computation of the minimal applications of \wedge I prior to any actual application of either \wedge I or \forall E will reveal that no such sat-equivalent formula ψ^* can be found. In the following, I will define such an equivalence transformation that will result in a formula ψ^* without the need to apply \wedge I in the case of a contradictory formula ψ . The ability to avoid the application of \wedge I is a significant feature of subformulae ψ . The ideal of an optimally \wedge I-minimal decision procedure can be realized in the case of such formulae.

To make this possible, however, one must use the following rule under certain conditions:

$$\exists\mu(\varphi(\mu) \wedge \psi(\mu)) \vdash \exists\mu_1\varphi(\mu/\mu_1) \wedge \exists\mu_2\psi(\mu/\mu_2) \quad \exists\text{M}$$

Table 4: Existential Quantifier Multiplication

We allow for indices of y variables at levels > 1 subsequent to the application of \exists M.

When an existential quantifier is multiplied by \exists M, the variable bound by the multiplied quantifier is ‘separated’ within the scope of the quantifier.

⁵Blocking restricts the introduction of new variables due to existential quantifier elimination in the tree calculus.

Definition 5.4. A variable is *separated* if it is subscripted with different indices at different positions subsequent to the application of $\exists M$ or other logical rules.

Hence, the application of $\exists M$ separates μ in the scope of $\exists\mu(\varphi(\mu) \wedge \psi(\mu))$ such that the occurrences of μ are subscripted with an index of 1 in the first conjunct and with an index of 2 in the second conjunct.

$\exists M$ can be applied at any level of the logical hierarchy of NNFs, like all other rules of the NNF-calculus. This rule must be applied in an optimally $\wedge I$ -minimal decision procedure for subformulae ψ . Such applications, however, must be restricted to those that are sat-equivalent. Section 7 specifies under what conditions this is the case. Section 8 proves that the contradictoriness of subformulae ψ can be decided within a $\wedge I$ -minimal proof strategy without $\exists M$ by applying $\wedge I$ no more than once. First, however, the following section 6 explains why and how the scopes in ψ must be minimized in the $\wedge I$ -minimal proof strategy.

6. MINIMIZING SCOPES

Given a subformula ψ in prenex normal form, the following conditions make it impossible to unify a pair of connected literals $\{L1, L2\}$ in ψ through the direct application of $\forall E$ without first applying equivalence transformations (that include $\wedge I$ or $\exists M$):

suboptimal prenex: An x variable ν must be replaced with a y variable μ to unify $\{L1, L2\}$, but $\exists\mu$ is in the scope of $\forall\nu$ in ψ .

x ambiguity: An x variable ν must be replaced with (at least) two existential y variables to unify $\{L1, L2\}$.

One cannot straightforwardly infer whether ψ is contradictory under these conditions. However, this situation changes if one considers equivalence transformations, in which minimizing scopes as much as possible is crucial.

| | | | |
|------------------------------------|----------------|--|------|
| $\forall\nu(A \wedge B(\nu))$ | $\dashv\vdash$ | $A \wedge \forall\nu B(\nu)$ | PN1 |
| $\forall\nu(B(\nu) \wedge A)$ | $\dashv\vdash$ | $\forall\nu B(\nu) \wedge A$ | PN2 |
| $\forall\nu(A \vee B(\nu))$ | $\dashv\vdash$ | $A \vee \forall\nu B(\nu)$ | PN3 |
| $\forall\nu(B(\nu) \vee A)$ | $\dashv\vdash$ | $\forall\nu B(\nu) \vee A$ | PN4 |
| $\exists\nu(A \wedge B(\nu))$ | $\dashv\vdash$ | $A \wedge \exists\nu B(\nu)$ | PN5 |
| $\exists\nu(B(\nu) \wedge A)$ | $\dashv\vdash$ | $\exists\nu B(\nu) \wedge A$ | PN6 |
| $\exists\nu(A \vee B(\nu))$ | $\dashv\vdash$ | $A \vee \exists\nu B(\nu)$ | PN7 |
| $\exists\nu(B(\nu) \vee A)$ | $\dashv\vdash$ | $\exists\nu B(\nu) \vee A$ | PN8 |
| $\forall\nu(A(\nu) \wedge B(\nu))$ | $\dashv\vdash$ | $\forall\nu A(\nu) \wedge \forall\nu B(\nu)$ | PN9 |
| $\exists\nu(A(\nu) \vee B(\nu))$ | $\dashv\vdash$ | $\exists\nu A(\nu) \vee \exists\nu B(\nu)$ | PN10 |

Table 5: PN Laws

Definition 6.1. *Anti-prenex normal forms* are NNFs with quantifiers that have been driven inward as far as possible through the application of PN laws; cf. table 5.⁶

Anti-prenex normal forms are a crucial part of a $\wedge I$ -minimal proof strategy to enable the computation of all possible *optimized prenex normal forms*; cf. section 10.

⁶ Cf. [Lampert(2017a)], section 2, for the history of anti-prenex normal forms and various algorithms for minimizing the scopes of quantifiers.

Definition 6.2. An *optimized prenex normal form* of NNF ϕ is a prenex normal form of ϕ in which the universal quantifiers have been brought within the scope of the existential quantifiers to the greatest possible extent by applying PN laws to generate prenex normal forms from an anti-prenex normal form of ϕ .

Optimized prenex normal forms are important for deciding whether there exists some *optimal* prenex, in which a universal quantifier $\forall\nu$ is within the scope of an existential quantifier $\exists\mu$ if ν is to be replaced with μ for unification; cf. p. 7 above and Definition 10.6 below.

Furthermore, within the \wedge I-minimal proof strategy, generating anti-prenex normal forms is important for multiplying universal quantifiers by applying PN9 to the maximal extent.⁷ This avoids the need to apply \wedge I in the case that x variables must be replaced with different y variables in different literals for unification. Minimizing scopes is a strategy for computing how many applications of \wedge I, if any, are necessary to generate optimal prenexes and to replace x variables in different literals with different y variables. Applications of \wedge I are at most necessary if, after transformation into anti-prenex normal forms, no optimal prenex can be generated or an x variable must be replaced with more than one y variable for unification.

Because ψ does not contain \vee , it is sufficient to apply PN1, PN2, PN5, PN6 and PN9 to generate the anti-prenex normal form ψ' from ψ by means of the following algorithm:

Algorithm 6.3.

- (1) $\psi = \psi'$.
- (2) If ψ' contains a subformula ρ in the form of $\forall\nu(A \wedge B(\nu))$, apply PN1 and replace ρ in ψ' with $A \wedge \forall\nu B(\nu)$. The new result becomes ψ' . Go to (2) if the new result ψ' is different from the initial formula ψ' ; otherwise, go to (3).
- (3) If ψ' contains a subformula ρ in the form of $\forall\nu(B(\nu) \wedge A)$, apply PN2 and replace ρ in ψ' with $\forall\nu(B(\nu) \wedge A)$. The new result becomes ψ' . Go to (2) if the new result ψ' is different from the initial formula ψ' ; otherwise, go to (4).
- (4) If ψ' contains a subformula ρ in the form of $\exists\nu(A \wedge B(\nu))$, apply PN5 and replace ρ in ψ' with $A \wedge \exists\nu B(\nu)$. The new result becomes ψ' . Go to (2) if the new result ψ' is different from the initial formula ψ' ; otherwise, go to (5).
- (5) If ψ' contains a subformula ρ in the form of $\exists\nu(B(\nu) \wedge A)$, apply PN6 and replace ρ in ψ' with $\exists\nu B(\nu) \wedge A$. The new result becomes ψ' . Go to (2) if the new result ψ' is different from the initial formula ψ' ; otherwise, go to (6).
- (6) If ψ' contains a subformula ρ in the form of $\forall\nu(A(\nu) \wedge B(\nu))$, apply PN9 and replace ρ in ψ' with $\forall\nu A(\nu) \wedge \forall\nu B(\nu)$. The new result becomes ψ' . Go to (2) if the new result ψ' is different from the initial formula ψ' ; otherwise, go to (7).
- (7) Maximally subscript ψ' , return the maximally subscripted result ψ' , and terminate.

When Algorithm 6.3 reaches (7), either no quantifier is present above \wedge in ψ' or ψ' contains a subformula ρ in the form of $\exists\nu(B(\nu) \wedge A(\nu))$.

⁷In the case of anti-prenex forms that contain \vee , the scope of universal quantifiers must be transformed into a CNF, and the scope of existential quantifiers must be transformed into a DNF under certain conditions. This results in further multiplications of quantifiers; cf. [Lampert(2017a)], section 2. In the case of a decision procedure for \wedge -NNFs, however, we can avoid such transformations.

Theorem 6.4. *Algorithm 6.3 is a logical equivalence transformation.*

Proof. Algorithm 6.3 involves nothing but the application of PN laws (cf. table 5) and laws of substitution (cf. table 1). These laws are all logical equivalence rules. \square

Since Algorithm 6.3 is a logical equivalence transformation, ψ' is a fortiori sat-equivalent to ψ .

Theorem 6.5. *Algorithm 6.3 terminates.*

Proof. This theorem follows from the fact that each rule reduces the depth of the formula, starting with a quantifier. If ψ contains n quantifiers, the algorithm will terminate after no more than n applications of one of the PN laws and $2 \cdot n$ subsequent applications of SUB1 or SUB2. \square

Let ψ' contain a sequence of existential quantifiers of length > 1 above \wedge , let the innermost existential quantifier of that sequence bind a variable that occurs in both conjuncts, and let the same not hold for all existential quantifiers of that sequence. In this case, sorting the quantifiers of the sequence such that all existential quantifiers that bind a variable occurring in only one conjunct are to the right of all other existential quantifiers in the sequence makes it possible to drive those quantifiers farther inward by applying PN5 or PN6. However, there is no need to consider such quantifier sorting here because it does not allow for any further application of PN1, PN2 or PN9, and it is only the application of these rules that leads to the intended effect of universal quantifiers either no longer being in the scope of certain existential quantifiers or being multiplied.

Example 6.6.

| No. | Formula | Rule |
|-----|---|--------|
| (1) | $\exists y_1 \forall x_1 \exists y_2 \forall x_2 \exists y_3 (F x_1 x_2 y_1 y_1 \wedge \neg F y_2 y_3 x_1 y_1)$ | ψ |
| (2) | $\exists y_1 \forall x_1 \exists y_2 \forall x_2 (F x_1 x_2 y_1 y_1 \wedge \exists y_3 \neg F y_2 y_3 x_1 y_1)$ | PN5 |
| (3) | $\exists y_1 \forall x_1 \exists y_2 (\forall x_2 F x_1 x_2 y_1 y_1 \wedge \exists y_3 \neg F y_2 y_3 x_1 y_1)$ | PN2 |
| (4) | $\exists y_1 \forall x_1 (\forall x_2 F x_1 x_2 y_1 y_1 \wedge \exists y_2 \exists y_3 \neg F y_2 y_3 x_1 y_1)$ | PN5 |
| (5) | $\exists y_1 (\forall x_1 \forall x_2 F x_1 x_2 y_1 y_1 \wedge \forall x_1 \exists y_2 \exists y_3 \neg F y_2 y_3 x_1 y_1)$ | PN9 |
| (6) | $\exists y_1 (\forall x_1 \forall x_2 F x_1 x_2 y_1 y_1 \wedge \forall x_3 \exists y_2 \exists y_3 \neg F y_2 y_3 x_3 y_1)$ | SUB2 |

Table 6: Transformation in Accordance with Algorithm 6.3

In step 1 of the sat-equivalence transformation that makes it possible to decide the contradictoriness of ψ , the subformula ψ is transformed into an anti-prenex formula ψ' . However, this is not sufficient to define a sat-equivalence transformation that drives quantifiers inward to the maximal extent. For this purpose, $\exists M$ must be applied to drive quantifiers above \wedge inward if they bind variables in both conjuncts and if such an application is sat-equivalent. The following section specifies the conditions for a sat-equivalent application of $\exists M$.

7. $\exists M$ -OPTIMIZATION

Applying $\exists M$ enables one to apply PN laws to further minimize the scopes of quantifiers.

Definition 7.1. $\exists M$ -*optimization* is a sat-equivalent application of $\exists M$ to an anti-prenex, maximally subscripted formula ψ' followed by the application of Algorithm 6.3.

The initial formula ψ must be transformed into a sat-equivalent, anti-prenex and $\exists M$ -optimized formula ψ'' to achieve proofs of contradiction without the need to apply $\wedge I$.

In the following, I will first identify two cases in which the application of $\exists M$ is not sat-equivalent. Then, I will prove that these are the only two cases in which the application of $\exists M$ is not sat-equivalent. In doing so, I will refer to nothing but the question of the unifiability of $\{L1, L2\}$ in ψ' . Thus, the sat-equivalent applications of $\exists M$ will be specified without referring to model theory or semantics.

What I call the ‘direct case’ of the violation of the requirements for a sat-equivalent application of $\exists M$ is defined below.

Definition 7.2. *Direct Case:* The y variable μ that is to be multiplied and, thus, separated by applying $\exists M$ occurs in identical positions in L1 and L2 before $\exists M$ is applied.

In this case, separating a y variable that occurs in identical positions before the application of $\exists M$ transforms a pair of connected literals into a pair of unconnected literals, which is no longer unifiable. This is already true in the simplest case of $\exists y_1(Fy_1 \wedge \neg Fy_1)$. This formula is contradictory. Applying $\exists M$ leads to the satisfiable (non-contradictory) formula $\exists y_1 Fy_1 \wedge \exists y_2 \neg Fy_2$. Therefore, the application of $\exists M$ is manifestly not sat-equivalent in the direct case. In this case, ‘ y unambiguity’ is lost: the same y variable occurs in identical positions before the application of $\exists M$, whereas different y variables occupy these positions after the application of $\exists M$. Hence, we exclude such cases from among the valid applications of $\exists M$.

The second case in which the requirements for a sat-equivalent application of $\exists M$ are violated is what I call the ‘indirect case’. To define it, let us introduce the concept of an ‘ xx list’, which we define recursively:

Definition 7.3. The xx lists of a pair of connected literals $\{L1, L2\}$ are generated as follows:

- (1) If an x variable ν_1 and an x variable ν_2 occur in identical positions in L1 and L2, then ν_1 and ν_2 are members of the same xx list.
- (2) If the x variables ν_1 and ν_2 are members of the same xx list and if ν_2 and the x variable ν_3 are also members of the same xx list, then ν_1 , ν_2 and ν_3 are all members of the same xx list.
- (3) (1) and (2) are the only conditions that determine members of the same xx list.

All x variables of an xx list must be replaced with the same y variable to unify $\{L1, L2\}$.

Example 7.4. Let ψ' be the following formula:

$$\exists y_1(\forall x_1 \forall x_2 Fy_1 x_1 x_2 x_2 \wedge \forall x_3 \forall x_4 \neg Fx_3 x_3 x_4 x_4 y_1) \quad (7.1)$$

The following pair of connected literals $\{L1, L2\}$ must be unified:

$$\{Fy_1 x_1 x_1 x_2 x_2, \neg Fx_3 x_3 x_4 x_4 y_1\}. \quad (7.2)$$

x_1 occurs in the same positions as x_3 and x_4 . Therefore, x_1 and x_3 as well as x_1 and x_4 are members of the same xx list, according to Definition 7.3(1). Hence, x_1 , x_3 and x_4 are also members of the same xx list, according to Definition 7.3(2). x_2 is in the same position as x_4 . Therefore, x_2 and x_4 are members of the same xx list, according to Definition 7.3(1). Since x_4 is also a member of the xx list $\{x_1, x_3, x_4\}$, x_2 is a member of this list as well,

according to Definition 7.3(2). This results in the following xx list, which is the only xx list of (7.2):

$$\{x_1, x_2, x_3, x_4\} \quad (7.3)$$

Furthermore, we define xy and yx pairs as follows:

Definition 7.5. An xy pair is an ordered list $\{xvar, yvar\}$ consisting of an x variable $xvar$ and a y variable $yvar$, where $xvar$ occurs in L1 in a position n and $yvar$ occurs in L2 in the same position n . A yx pair is an ordered list $\{yvar, xvar\}$ consisting of a y variable $yvar$ and an x variable $xvar$, where $yvar$ occurs in L1 in a position n and $xvar$ occurs in L2 in the same position n .

An x variable $xvar$ must be replaced with the y variable $yvar$ if $xvar$ and $yvar$ occur in the same xy or yx pair.

Example 7.6. (7.4) is the only xy pair of (7.2), and (7.5) is the only yx -pair of (7.2).

$$\{x_2, y_1\} \quad (7.4)$$

$$\{y_1, x_3\} \quad (7.5)$$

(7.1) is contradictory: (7.2) can be unified by replacing all x variables with y_1 , which is possible because $\{y_1, x_1, x_2, x_3, x_4\}$ represents an optimal prenex that can be generated by applying PN laws to (7.1). The application of $\exists M$ to (7.1), however, is not sat-equivalent because this would separate y_1 into y_{1_1} in the first conjunct and y_{1_2} in the second conjunct. Consequently, x_3 must be replaced with y_{1_1} because of (7.5), and x_2 must be replaced with y_{1_2} because of (7.4). However, according to (7.3), x_1, x_2, x_3 and x_4 all must be replaced with the same y variable to unify (7.2).

Based on the above discussion, the indirect case of the violation of the requirements for a sat-equivalent application of $\exists M$ can be defined as follows:

Definition 7.7. *Indirect case:* $\{L1, L2\}$ has an xx list A such that at least one x variable $xvar1$ from A is a member of an xy list $\{xvar1, yvar\}$, at least one x variable $xvar2$ from A ($xvar1 \neq xvar2$) is a member of a yx list $\{yvar, xvar2\}$, and all universal quantifiers binding the x variables in A are within the scope of $\exists yvar$ in ψ , where $yvar$ is the y variable multiplied by $\exists M$.

On p. 17 below, I will justify why this case is restricted to x variables that are bound by quantifiers in the scope of the existential quantifier that is multiplied by $\exists M$ (Case 3(β) below). In contrast to the direct case, one might say that ‘ x unambiguity’ is lost in the indirect case: whereas all x variables of an xx list must be replaced with one and the same y variable before the application of $\exists M$, they must be replaced with different y variables after the application of $\exists M$. Hence, we exclude not only the direct case but also the indirect case from among the valid applications of $\exists M$.

In the following, I justify why the direct and indirect cases are the only two cases that must be excluded to restrict $\exists M$ to sat-equivalent applications. In doing so, we consider nothing but logically valid equivalence transformations to unify positions of a pair of connected literals within a subformula ψ' .

The syntactic properties that determine the unifiability of a pair of connected literals are (i) the order of the quantifiers and (ii) the identity or difference of the variables at certain positions of the literals. The order of the quantifiers, however, is not adversely affected by the application of $\exists M$ because any $\exists M$ -optimization results in a minimized scope. This can

only increase the possibility of generating optimal prenexes. In addition to the changes to quantifier order, $\exists M$ separates the multiplied y variable as well as those x variables that (i) are bound by a universal quantifier that is to the left of the multiplied existential quantifier before $\exists M$ is applied and (ii) occur in both literals, L1 and L2. The separation of x variables subsequent to the separation of the multiplied y variable is due to the application of PN9 in $\exists M$ -optimization. This separation cannot have a negative effect on unifiability because it enables the coverage of positions in different literals with different y variables. Therefore, the unifiability of a pair of literals can only be adversely affected by the separation of the multiplied y variable, which is subscripted differently in the two conjuncts subsequent to $\exists M$ -optimization.

To identify the cases in which the separation of the multiplied y variable μ transforms a unifiable pair of connected literals into a pair of literals that is no longer unifiable, we distinguish three cases:

Case 1: μ occurs in identical positions in L1 and L2.

Case 2: μ occurs in identical positions to those of x variables that are not members of an xx list.

Case 3: μ occurs in identical positions to those of x variables that are members of an xx list.

Note that there is no need to distinguish a case in which μ occurs in the same position as some y variable that is not identical to μ because this violates condition (4) of the definition of a pair of connected literals; cf. Definition 2.2, p. 3. Therefore, these three cases are exhaustive. It may happen that the conditions for more than one of the three cases are satisfied for one and the same pair of connected literals. However, the three cases can be considered separately because the application of $\exists M$ is not sat-equivalent iff the multiplication of μ does not preserve unifiability in regard to at least one case.

Case 1 is the direct case. Case 2 cannot induce applications of $\exists M$ that are not sat-equivalent, as illustrated by the following reasoning: The x variables need to be replaced with μ either (α) in only one of the two literals or (β) in both. In the first case (α), separation of the y variable cannot adversely affect the possibility of unifying the positions in which the x variables occur because the number of y variables with which those x variables need to be replaced for unification does not change. The second case (β) requires that the universal quantifiers that bind x variables that occur in identical positions to those of μ must be to the left of $\exists\mu$ before the application of $\exists M$ because otherwise, the x variables will not occur in both L1 and L2 due to the presumed maximal subscription. However, given that (a) $\exists\mu$ is in the scope of a universal quantifier $\forall\nu$ in the maximally subscripted formula ψ' and (b) ν and μ occur in both L1 and L2 in identical positions (and ν , therefore, must be replaced with μ in both literals), L1 and L2 are not unifiable anyway, and consequently, ψ' is not contradictory. ψ' is not contradictory because it is impossible in case (β) to generate an optimal prenex that would permit the necessary substitutions, even after the minimization of scopes through $\exists M$ -optimization or the application of $\wedge I$ (which of the two is done makes no difference; cf. section 8). In any case, at least one universal quantifier $\forall\nu_i$ ($\nu_i = \nu$ with an additional subscript of 1 or 2) remains that is necessarily not in the scope of $\exists\mu_j$ ($\mu_i = \mu$ with an additional subscript of 1 or 2) in any optimized prenex.

The indirect case is a special case of case 3. Again, one can ignore a case (α) in which there is either no xy pair that contains both an x variable from an xx list and μ as members or no yx pair that contains both an x variable from an xx list and μ as members. In such a case, the number of necessary substitutions in regard to the x variables of the xx list would

again not be increased by applying $\exists M$. However, in a case (β) in which some x variable $xvar1$ in an xx list occurs together with μ in an xy pair and some x variable $xvar2$ in the same xx list occurs together with μ in a yx pair, the situation is different. Nevertheless, we can identify several conditions under which the application of $\exists M$ is still sat-equivalent. First of all, we can assert that $xvar1 \neq xvar2$ because otherwise the case reduces to case 2. Furthermore, if one universal quantifier $\forall \nu$ that binds an x variable ν from the xx list is to the left of $\exists \mu$ in ψ' , then $\{L1, L2\}$ is not unifiable, and ψ' is not contradictory. This is so because in this case, minimizing scopes through $\exists M$ -optimization or the application of $\wedge I$ is necessary. However, minimizing scopes or applying $\wedge I$, where both include maximal subscription, makes it necessary for at least one x variable from the xx list to be replaced with both μ_1 and μ_2 for unification in one and the same literal, which is logically impossible. Therefore, the only subcase of case 3 is the indirect case, in which $xvar1 \neq xvar2$ and all universal quantifiers binding x variables from an xx list are in the scope of $\exists \mu$ in ψ' .

Thus, the direct and indirect cases are the only cases that must be excluded to restrict the application of $\exists M$ to sat-equivalent transformations.

Definition 7.8. $\exists M_B$ is the rule stating that $\exists M$ is applied unless the conditions for either the direct or the indirect case are satisfied.

Theorem 7.9. *The application of $\exists M_B$ is sat-equivalent.*

Proof. This theorem follows from the above distinction of cases, which shows that the direct and indirect cases are the only cases that imply an application of $\exists M$ that is not sat-equivalent. The proof is based on nothing but proof-theoretic considerations concerning the unification of the pair of connected literals in ψ' in the NNF-calculus. \square

The following algorithm transforms the initial subformula ψ into a sat-equivalent, anti-prenex and $\exists M$ -optimized subformula ψ'' :

Algorithm 7.10.

- (1) Apply Algorithm 6.3 to ψ . The result is ψ' .
- (2) If an existential quantifier exists above \wedge , apply $\exists M_B$ to multiply this quantifier if possible. The result is ψ'' .
- (3) If $\psi' = \psi''$, return ψ'' and terminate; otherwise, go back to (1).

Theorem 7.11. *Algorithm 7.10 is a sat-equivalence transformation.*

Proof. This theorem follows from Theorems 6.4 and 7.9. \square

Theorem 7.12. *Algorithm 7.10 terminates.*

Proof. This theorem follows from Theorem 6.5 and the fact that there can be only a finite number of existential quantifiers in the logical hierarchy above \wedge in ψ . \square

Theorem 7.13. *Algorithm 7.10 minimizes the scope of universal quantifiers in ψ to a maximal extent under equivalence transformations and thus maximizes the number of universal quantifiers binding the variables occurring in a pair of connected literals to a maximal extent under equivalence transformations.*

Proof. The scope of universal quantifiers is not minimized any further by Algorithm 7.10 only if an existential quantifier exists that is in their scope and above \wedge and if the multiplication of this quantifier is not sat-equivalent. Minimizing scopes maximizes the number of applications of PN9 and, therefore, the number of universal quantifiers. \square

Either the resulting formula ψ'' is a conjunction of two quantified expressions, each one containing exactly one literal, or ψ'' is a quantified expression with an outermost sequence of quantifiers ending with an existential quantifier $\exists\mu$ above \wedge , where each conjunct contains exactly one literal and the conditions for either the direct or the indirect case are satisfied.

8. $\exists\text{M}$ -OPTIMIZATION AND THE APPLICATION OF $\wedge\text{I}$

$\exists\text{M}$ -optimization minimizes scopes and separates variables. This makes applications of $\wedge\text{I}$ superfluous in the case of subformulae ψ' . This section demonstrates this claim, first by example and then in general.

Example 8.1. Let ψ' be the following formula:

$$\forall x_1 \exists y_1 (\forall x_3 Fy_1 x_1 x_3 \wedge \forall x_2 \neg Fx_2 y_1 y_1) \quad (8.1)$$

x_1 must be replaced with y_1 to unify $\{Fy_1 x_1 x_3, \neg Fx_2 y_1 y_1\}$. However, $\forall\text{E}$ cannot be applied to (8.1) to achieve this because $\exists y_1$ is in the scope of $\forall x_1$. Independent of $\exists\text{M}$ -optimization, one must apply $\wedge\text{I}$ to replace a derivate of x_1 with a derivate of y_1 ; cf. Definition 4.2. Applying $\wedge\text{I}$ once to multiply $\forall x_1$ results in the following formula:

$$\forall x_1 \exists y_1 (\forall x_3 Fy_1 x_1 x_3 \wedge \forall x_2 \neg Fx_2 y_1 y_1) \wedge \quad (8.2a)$$

$$\forall x_1 \exists y_1 (\forall x_3 Fy_1 x_1 x_3 \wedge \forall x_2 \neg Fx_2 y_1 y_1) \wedge \quad (8.2b)$$

This formula allows one to derive an explicit contradiction by unifying the pair $\{Fy_1 x_1 x_3, \neg Fx_2 y_1 y_1\}$ through the application of $\forall\text{E}$ to a corresponding optimal prenex normal form. Since unifying the stated pair of literals is sufficient to derive an explicit contradiction, we delete the other two literals:

$$\forall x_1 \exists y_1 \forall x_3 Fy_1 x_1 x_3 \wedge \exists y_1 \forall x_2 \neg Fx_2 y_1 y_1 \quad (8.3)$$

Applying PN laws allows one to generate the following optimal prenex normal form from (8.3):

$$\exists y_1 \forall x_1 \exists y_1 \forall x_3 \forall x_2 (Fy_1 x_1 x_3 \wedge \neg Fx_2 y_1 y_1) \quad (8.4)$$

This makes it possible to apply $\forall\text{E}$ such that the pair of literals in (8.4) is unified by the following substitutions:

$$\{\{x_1, y_1\}, \{x_2, y_1\}, \{x_3, y_1\}\}. \quad (8.5)$$

However, the application of $\wedge\text{I}$ that is involved in this proof is apparently only a cumbersome surrogate for a corresponding $\exists\text{M}$ -optimization. This is so because $\exists\text{M}$ -optimization directly leads to the following formula, which is identical to (8.3) despite the naming of the x variables:

$$\forall x_1 \exists y_1 \forall x_3 Fy_1 x_1 x_3 \wedge \exists y_1 \forall x_2 \neg Fx_2 y_1 y_1 \quad (8.6)$$

Theorem 8.2. *$\exists\text{M}$ -optimization makes the application of $\wedge\text{I}$ superfluous in proofs of contradiction for subformulae ψ .*

Proof. Applications of $\wedge\text{I}$ are necessary for unification as long as one does not refer to $\exists\text{M}$ -optimized formulae ψ'' in the following cases:

- (1) to replace a universal variable ν with an existential variable μ , where $\exists\mu$ is in the scope of $\forall\nu$ in ψ (case 1; cf. ‘suboptimal prenex’, p. 11), or

- (2) to separate a universal variable ν in ψ such that the different variables ν_1 and ν_2 can be replaced with different y variables (case 2; cf. ‘ x ambiguity’, p. 11).

However, these substitutions can be achieved through $\exists M$ -optimization as long as the substitutions are logically valid. The conditions for case 1 or case 2 only hold in $\exists M$ -optimized expressions ψ'' if $\exists\rho$ is in the scope of $\forall\nu$ and above \wedge and if multiplying $\exists\mu$ is incompatible with unifying the pair of literals in ψ' . Any application of $\wedge I$ to $\forall\nu$ would necessarily also multiply $\exists\rho$. Therefore, applications of $\wedge I$ cannot enable unifications that cannot be achieved through $\exists M$ -optimization. \square

Example 8.3. The following formulae exemplify the two cases described above:

$$\forall x_1 \exists y_1 (\exists y_2 F y_1 y_2 \wedge \neg F y_1 x_1) \quad (8.7)$$

$$\exists y_1 \exists y_2 \forall x_1 \exists y_3 (\exists y_2 F y_3 y_1 x_1 \wedge \neg F y_3 x_1 y_2) \quad (8.8)$$

In (8.7), $\nu = x_1$, $\rho = y_1$, and $\mu = y_2$; in (8.8), $\nu = x_1$, $\mu_1 = y_1$, $\mu_2 = y_2$, and $\rho = y_3$. To replace x_1 in (8.7) with y_1 , or to replace x_1 in L1 of (8.8) with x_{1_1} and in L2 of (8.8) with x_{1_2} such that x_{1_1} can be replaced with y_2 and x_{1_2} with y_1 , one must multiply $\exists y_1$ in (8.7) and $\exists y_3$ in (8.8). This applies to both $\exists M$ -optimization and the application of $\wedge I$. However, in both cases, this is incompatible with unifying the first position of both literals.

Theorem 8.4. *One can generate ψ'' from ψ without applying $\exists M$ by applying $\wedge I$ once.*

Proof. Let $\exists\mu$ be the outermost existential quantifier that is multiplied in Algorithm 7.10. Then, one can obtain ψ'' from ψ by applying the following algorithm:

- (1) Apply Algorithm 6.3 to minimize scopes and rename variables.
- (2) Multiply $\exists\mu A(\mu)$ by applying $\wedge I$ once.
- (3) Delete L2 from the first conjunct along with all quantifiers binding variables that occur only in L2.
- (4) Delete L1 from the second conjunct along with all quantifiers binding variables that occur only in L1.
- (5) Apply Algorithm 6.3 to minimize scopes and rename variables.

\square

Thus, in a $\wedge I$ -minimal proof strategy for subformulae ψ in the NNF-calculus without $\exists M$, $\wedge I$ will be applied at most once.

To decide whether ψ is contradictory, it is sufficient to transform ψ into a sat-equivalent, $\exists M$ -optimized formula ψ'' ; cf. section 13. In the following, I will first describe how to compute the substitutions that are necessary for unifying the pairs of literals in ψ'' (section 9). Afterward, I will describe how to generate all optimized prenex normal forms ψ''' from ψ'' (section 10).

9. SUBSTITUTION LIST σ

To decide whether ψ is contradictory, a substitution list σ is generated. For each x variable, the list specifies the y variables with which it must be replaced to unify the pair of connected literals $\{L1, L2\}$ in ψ'' . Such a list does not imply that $\{L1, L2\}$ is unifiable. It may well be that some x variables would need to be replaced with several y variables to unify $\{L1, L2\}$. The following algorithm generates σ for a given pair of literals $\{L1, L2\}$:

Algorithm 9.1.

- (1) Generate a list $\sigma 1$ of all pairs of variables $\{v_1, v_2\}$ with v_1 in position n in $L1$ and v_2 in the same position n in $L2$, where v_1 or v_2 (or both) is an x variable.
- (2) To each level-1 list that contains an x variable ν_1 and exactly one y variable μ , recursively add those x variables ν_2 ($\nu_1 \neq \nu_2$) that occur in some other level-1 list together with ν_1 . Delete any level-1 list that contains only x variables if all of those x variables also occur in some level-1 list that contains a y variable. Repeat this procedure until no further changes occur. The resulting list is denoted by $\sigma 2$.
- (3) Generate a list $\sigma 3$ consisting of level-1 lists that each contain one x variable ν and all y variables that occur together with ν in some level-1 list in $\sigma 2$. To generate the level-1 lists in $\sigma 3$, traverse all x variables ν that occur in $\sigma 2$ and take all y variables that are members of lists in $\sigma 2$ of which ν is also a member. If ν is not a member of any list in $\sigma 2$ that contains a y variable, add $\{\nu, y_0\}$ to $\sigma 3$. Per its definition, y_0 does not occur in ψ , ψ' or ψ'' . The resulting list is the substitution list σ that specifies how each x variable is to be substituted such that $\{L1, L2\}$ is unified.

Example 9.2. Let $\{L1, L2\}$ be the following pair:

$$\{Fx_1x_2x_2x_4, \neg Fy_2x_1x_3x_4\} \quad (9.1)$$

- $\sigma 1$: $\{\{x_1, y_2\}, \{x_2, x_1\}, \{x_2, x_3\}, \{x_4, x_4\}\}$.
 $\sigma 2$: $\{\{x_1, x_2, x_3, y_2\}, \{x_4, x_4\}\}$.
 $\sigma 3$ ($=\sigma(9.1)$): $\{\{x_1, y_2\}, \{x_2, y_2\}, \{x_3, y_2\}, \{x_4, y_0\}\}$.

Definition 9.3. x lists are the members (= level-1 lists) of the substitution list σ .

Definition 9.4. An x list is *unambiguous* iff it contains exactly one y variable. σ is *unambiguous* iff all x lists in σ are unambiguous.

Example 9.5. The x lists of $\sigma 3$ in Example 9.2 and, therefore, $\sigma(9.1)$ are unambiguous.

Definition 9.6. A substitution list σ is *maximally separated* iff the number of members in σ cannot be increased any further by separating x variables through equivalence transformations.

Since Algorithm 7.10 minimizes the scopes of universal quantifiers to the maximal extent under equivalence transformation and, thus, increases the number of universal quantifiers to the maximal extent (cf. Theorem 7.13), the substitution list of ψ'' is maximally separated. Unlike the substitution list of ψ , the substitution list of ψ'' is necessarily maximally separated but is not necessarily unambiguous.

Example 9.7. Let ψ'' be as follows:

$$\exists y_1 \exists y_2 \forall x_1 \exists y_3 (Fy_1x_1y_3 \wedge \neg Fx_1y_2y_3) \quad (9.2)$$

The corresponding substitution list is $\sigma(9.2) = \{\{x_1, y_1, y_2\}\}$. This substitution list is not unambiguous. However, $\exists M$ cannot be applied to (9.2) because the conditions for the direct case are satisfied. Hence, $\sigma(9.2)$ is maximally separated but not unambiguous.

A decision on the unifiability of a pair of connected literals refers to a *maximally separated* substitution list. A proof of contradiction according to the $\wedge I$ -minimal proof strategy in the NNF-calculus, however, requires an *unambiguous* substitution list.

Algorithm 9.1 generates not *minimal* but *maximal* substitution lists. In $\{Fx_1x_1y_1, \neg Fx_2y_2x_2\}$, for example, $\sigma = \{\{x_1, y_1, y_2\}, \{x_2, y_1, y_2\}\}$: both x lists contain both y_1 and y_2 , which is not necessary. It is only necessary that x_1 be substituted with y_2 in position 2 of L1, that x_2 be substituted with y_1 in position 3 of L2, and that x_1 and x_2 be substituted with one and the same y variable in position 1 of both L1 and L2. This last condition is compatible with two different *minimal* substitution lists: $\{\{x_1, y_1, y_2\}, \{x_2, y_1\}\}$ or, alternatively, $\{\{x_1, y_2\}, \{x_2, y_1, y_2\}\}$. However, when deciding whether subformulae ψ are contradictory, one can abstain from generating alternative minimal substitution lists. It is sufficient to refer to a maximal substitution list σ because – as proven in section 11 – a necessary condition for the contradiction of a subformula ψ is that the σ of ψ'' be unambiguous, which implies that there is only one minimal substitution list in this case. If the maximal substitution list σ is not unambiguous, this implies that *at least one* x variable must be replaced with more than one y variable to unify the pair of literals in ψ'' . This is sufficient to conclude that ψ'' and, therefore, ψ are not contradictory; cf. Theorem 11.1, condition C1, p. 24.

Section 13 will explain how it can be decided whether the pair of literals contained in ψ'' is unifiable by referring to the syntactic properties of ψ'' (without any further equivalence transformation). In the following section, however, I will describe a costlier algorithm for such decision-making. This algorithm generates the set of all optimized prenex normal forms ψ''' from the anti-prenex, \exists M-optimized normal form ψ'' . I present this algorithm first because it serves as the foundation for the syntactic criteria defined in section 13 that directly refer to ψ'' .

10. OPTIMIZED PRENEXES

This section describes an algorithm that generates all optimized prenex normal forms (cf. Definition 6.2) from an anti-prenex, \exists M-optimized and maximally subscripted formula ψ'' . Since ψ'' does not contain \vee , only PN1, PN2, PN5 and PN6 need to be considered.

To generate the set of all optimized prenex normal forms ψ''' from ψ'' by applying PN laws from right to left, proceed as follows:

Algorithm 10.1.

- (1) $\psi^+ = \{\psi''\}$.
- (2) $\psi''' = \{\}$.
- (3) If $\psi^+ \neq \{\}$, select the first member ψ_1^+ from ψ^+ and go to (a); otherwise, return ψ''' and terminate.
 - (a) Let ψ_1^+ be of the form $Q(A \wedge B)$, where Q is a sequence of quantifiers of length m ($m \geq 0$). If neither A nor B contains any quantifier, add ψ_1^+ to ψ''' , delete ψ_1^+ in ψ^+ and go to (3); otherwise, go to (b).
 - (b) If PN6 can be applied to $A \wedge B$, then $\psi_1^+ = QPN6[A \wedge B]$ and go to (a); otherwise, go to (c).
 - (c) If PN5 can be applied to $A \wedge B$, then $\psi_1^+ = QPN5[A \wedge B]$ and go to (a); otherwise, go to (d).
 - (d) If A contains only universal quantifiers, then $\psi_1^+ = QPN1[A \wedge B]$ and go to (a); otherwise, go to (e).
 - (e) If B contains only universal quantifiers, then $\psi_1^+ = QPN2[A \wedge B]$ and go to (a); otherwise, go to (f).

(f) Delete ψ_1^+ in ψ^+ and add $QPN2[A \wedge B]$ and $QPN1[A \wedge B]$ to ψ^+ . Go to (3).

Example 10.2. Let ψ'' be as follows:

$$\exists y_1 \forall x_1 \exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \forall x_3 \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5 \quad (10.1)$$

The following schema illustrates the application of Algorithm (10.1) to generate ψ^+ and ψ''' from (10.1).

| No. | $\psi^+; \psi'''$ | Rule |
|-----|--|------|
| (1) | $\psi^+ = \{ \exists y_1 \forall x_1 \exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \forall x_3 \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5 \};$ $\psi''' = \{ \}$ | 1, 2 |
| (2) | $\psi^+ = \{ \exists y_1 (\forall x_1 \exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \forall x_3 \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5) \};$ $\psi''' = \{ \}$ | 3(b) |
| (3) | $\psi^+ = \{ \exists y_1 \forall x_1 (\exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \forall x_3 \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5),$ $\exists y_1 \forall x_3 (\forall x_1 \exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5) \};$ $\psi''' = \{ \}$ | 3(f) |
| (4) | $\psi^+ = \{ \exists y_1 \forall x_1 \exists y_2 (\forall x_2 F y_1 x_1 y_2 x_2 \wedge \forall x_3 \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5),$ $\exists y_1 \forall x_3 (\forall x_1 \exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5) \};$ $\psi''' = \{ \}$ | 3(b) |
| (5) | $\psi^+ = \{ \exists y_1 \forall x_1 \exists y_2 \forall x_3 (\forall x_2 F y_1 x_1 y_2 x_2 \wedge \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5),$ $\exists y_1 \forall x_3 (\forall x_1 \exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5) \};$ $\psi''' = \{ \}$ | 3(d) |
| (6) | $\psi^+ = \{ \exists y_1 \forall x_1 \exists y_2 \forall x_3 \exists y_3 (\forall x_2 F y_1 x_1 y_2 x_2 \wedge \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5),$ $\exists y_1 \forall x_3 (\forall x_1 \exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5) \};$ $\psi''' = \{ \}$ | 3(c) |
| (7) | $\psi^+ = \{ \exists y_1 \forall x_1 \exists y_2 \forall x_3 \exists y_3 \forall x_4 (\forall x_2 F y_1 x_1 y_2 x_2 \wedge \forall x_5 \neg F x_3 y_3 x_4 x_5),$ $\exists y_1 \forall x_3 (\forall x_1 \exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5) \};$ $\psi''' = \{ \}$ | 3(d) |
| (8) | $\psi^+ = \{ \exists y_1 \forall x_1 \exists y_2 \forall x_3 \exists y_3 \forall x_4 \forall x_5 (\forall x_2 F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5),$ $\exists y_1 \forall x_3 (\forall x_1 \exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5) \};$ $\psi''' = \{ \}$ | 3(d) |
| (9) | $\psi^+ = \{ \exists y_1 \forall x_1 \exists y_2 \forall x_3 \exists y_3 \forall x_4 \forall x_5 \forall x_2 (F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5),$ $\exists y_1 \forall x_3 (\forall x_1 \exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5) \};$ $\psi''' = \{ \}$ | 3(e) |

| | | |
|------|---|------|
| (10) | $\psi^+ = \{\exists y_1 \forall x_3 (\forall x_1 \exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \exists y_3 \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5)\};$ $\psi''' = \{\exists y_1 \forall x_1 \exists y_2 \forall x_3 \exists y_3 \forall x_4 \forall x_5 \forall x_2 (F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5)\}$ | 3(a) |
| (11) | $\psi^+ = \{\exists y_1 \forall x_3 \exists y_3 (\forall x_1 \exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5)\};$ $\psi''' = \{\exists y_1 \forall x_1 \exists y_2 \forall x_3 \exists y_3 \forall x_4 \forall x_5 \forall x_2 (F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5)\}$ | 3(c) |
| (12) | $\psi^+ = \{\exists y_1 \forall x_3 \exists y_3 \forall x_1 (\exists y_2 \forall x_2 F y_1 x_1 y_2 x_2 \wedge \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5)\};$ $\psi''' = \{\exists y_1 \forall x_1 \exists y_2 \forall x_3 \exists y_3 \forall x_4 \forall x_5 \forall x_2 (F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5)\}$ | 3(e) |
| (13) | $\psi^+ = \{\exists y_1 \forall x_3 \exists y_3 \forall x_1 \exists y_2 (\forall x_2 F y_1 x_1 y_2 x_2 \wedge \forall x_4 \forall x_5 \neg F x_3 y_3 x_4 x_5)\};$ $\psi''' = \{\exists y_1 \forall x_1 \exists y_2 \forall x_3 \exists y_3 \forall x_4 \forall x_5 \forall x_2 (F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5)\}$ | 3(b) |
| (14) | $\psi^+ = \{\exists y_1 \forall x_3 \exists y_3 \forall x_1 \exists y_2 \forall x_4 (\forall x_2 F y_1 x_1 y_2 x_2 \wedge \forall x_5 \neg F x_3 y_3 x_4 x_5)\};$ $\psi''' = \{\exists y_1 \forall x_1 \exists y_2 \forall x_3 \exists y_3 \forall x_4 \forall x_5 \forall x_2 (F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5)\}$ | 3(d) |
| (15) | $\psi^+ = \{\exists y_1 \forall x_3 \exists y_3 \forall x_1 \exists y_2 \forall x_4 \forall x_5 (\forall x_2 F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5)\};$ $\psi''' = \{\exists y_1 \forall x_1 \exists y_2 \forall x_3 \exists y_3 \forall x_4 \forall x_5 \forall x_2 (F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5)\}$ | 3(d) |
| (16) | $\psi^+ = \{\exists y_1 \forall x_3 \exists y_3 \forall x_1 \exists y_2 \forall x_4 \forall x_5 \forall x_2 (F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5)\};$ $\psi''' = \{\exists y_1 \forall x_1 \exists y_2 \forall x_3 \exists y_3 \forall x_4 \forall x_5 \forall x_2 (F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5)\}$ | 3(e) |
| (17) | $\psi^+ = \{\};$ $\psi''' = \{\exists y_1 \forall x_1 \exists y_2 \forall x_3 \exists y_3 \forall x_4 \forall x_5 \forall x_2 (F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5),$ $\exists y_1 \forall x_3 \exists y_3 \forall x_1 \exists y_2 \forall x_4 \forall x_5 \forall x_2 (F y_1 x_1 y_2 x_2 \wedge \neg F x_3 y_3 x_4 x_5)\}$ | 3(a) |

Theorem 10.3. *Algorithm 10.1 generates only prenex normal forms that are logically equivalent to ψ'' .*

Proof. This theorem follows from the fact that the PN laws are equivalence rules that are applied from right to left in Algorithm 10.1. \square

Theorem 10.4. *Algorithm 10.1 terminates.*

Proof. This theorem follows from the fact that only a finite number of quantifiers can exist below \wedge in the logical hierarchy of ψ'' . \square

Theorem 10.5. *Algorithm 10.1 generates the set of all possible optimized prenex normal forms ψ''' from ψ'' .*

Proof. This theorem follows from the facts that in Algorithm 10.1, PN5 and PN6 are applied prior to PN1 and PN2, and two alternative prenex forms are generated in the case of 3(g). \square

Definition 10.6. An *optimal* prenex normal form ψ^* of ψ'' is a prenex normal form in ψ''' such that for all x lists $\{\nu, \dots, \mu_j, \dots\}$ ($j > 0$), $\forall \nu$ is to the right of each existential quantifier $\exists \mu_j$.

Example 10.7. $\sigma(10.1): \{\{x_1, y_3\}, \{x_2, y_0\}, \{x_3, y_1\}, \{x_4, y_2\}, \{x_5, y_0\}\}.$

Hence, only the second prenex normal form in ψ''' from line (17) in the schema above is optimal.

11. SAT-CRITERIA FOR ψ

Theorem 11.1. *ψ is not contradictory iff one of the two following conditions is satisfied:*

- C1:** *The maximally separated substitution list σ of the pair of connected literals in ψ'' (and, consequently, of each formula from ψ''') is ambiguous.*
C2: *No optimal prenex normal form ψ^* exists in ψ''' .*

C1 concerns the substitution list σ of ψ'' , whereas C2 concerns the optimized prenexes in ψ''' . These are the relevant features on which the unifiability of the pair of connected literals in ψ'' depends. The unifiability of a pair of literals can be decided by generating ψ , ψ' , ψ'' , ψ''' , and σ to determine whether C1 or C2 is satisfied.

Proof. We prove that Theorem 11.1 holds in the direction from left to right by proving the contrapositive: ‘If neither C1 nor C2 is satisfied, then ψ is contradictory’. This follows from the following two points:

- (1) If C2 is not satisfied, then some optimal prenex form ψ^* exists, and if C1 is not satisfied, then some unambiguous substitution list σ exists. Under such conditions, the pair of connected literals in ψ^* can be unified by applying $\forall E$ to replace each x variable in σ with the y variable in the corresponding x list of σ .
- (2) The transformation of ψ into ψ^* is a sat-equivalence transformation according to Theorems 7.11 and 10.3.

□

In the proof of the right-to-left direction of Theorem 11.1, C1 and C2 are reduced to ‘laws of invalidity’ that specify syntactic properties that prevent the application of $\forall E$ for unification. The all-important point is that one can conclude from this that unification is impossible if one refers to optimized prenex normal forms in ψ''' . The reason for this conclusion is that the conditions for replacing x variables with y variables when applying $\forall E$ are optimal in the case of the optimized prenex normal forms of ψ''' . Therefore, one can conclude that *no* sat-equivalent prenex normal form exists that makes the substitutions that are necessary for unification according to σ logically possible if no prenex normal form exists in ψ''' that permits such a substitution.

Proof. We prove that Theorem 11.1 holds in the direction from right to left: ‘If C1 or C2 is satisfied, then ψ is not contradictory.’ According to Theorems 7.11, 10.3 and 10.5, ψ''' contains all optimized prenex normal forms that are sat-equivalent to ψ . It is necessary to show that C1 and C2 specify the conditions for ψ to be non-contradictory.

If the maximally separated substitution list σ is ambiguous, then at least one x variable in ψ'' and in each formula in ψ''' must be replaced with more than one y variable. However, this is logically impossible. C1 is based on the following law of invalidity:

$$\dots \exists \mu_1 \dots \exists \mu_2 \dots \forall \nu \varphi(\mu_1, \mu_2, \nu, \nu) \not\vdash \dots \exists \mu_1 \dots \exists \mu_2 \dots \varphi(\mu_1, \mu_2, \nu/\mu_1, \nu/\mu_2) \quad \text{C1-R.}$$

$\varphi(\mu_1, \mu_2, \nu, \nu)$ is an expression in which ν must be replaced with μ_1 and also with μ_2 to unify the pair of literals contained in the expression. However, such a substitution is logically invalid because each application of $\forall E$ can only replace ν with either μ_1 or μ_2 and, according to Theorems 7.13 and 8.2, the number of universal quantifiers cannot be increased any further for the purpose of unification. It is impossible to separate σ any further through sat-equivalence transformations because σ is already maximally separated. However, σ would need to be further separated to make it possible to unify the pair of literals in question by applying $\forall E$.

C2 is based on the following law of invalidity for the set of optimized prenex normal forms ψ''' :

$$\dots \forall \nu \dots \exists \mu \varphi(\mu, \nu) \not\vdash \dots \exists \mu \dots \varphi(\mu, \nu/\mu) \quad \text{B2-R.}$$

The application of $\forall E$ requires that $\forall \nu$ be in the scope of $\exists \mu$. According to Theorem 10.5, Algorithm 10.1 generates *all* optimized prenex normal forms ($= \psi'''$). The quantifier sequences of the prenex normal forms in ψ''' cannot be optimized any further through satisfiability transformations, according to Theorems 7.13 and 8.2. If no optimized prenex normal form in ψ''' permits the application of $\forall E$ to achieve all substitutions that are necessary according to σ , then no optimal prenex normal form ψ^* exists that enables one to prove that ψ is contradictory. \square

Example 11.2. The following paradigmatic examples of anti-prenex, $\exists M$ -optimized formulae ψ'' cannot be transformed into explicit contradictions because of condition C1:

$$\exists y_1 \exists y_2 F y_1 y_2 \wedge \forall x_1 \neg F x_1 x_1 \not\vdash \exists y_1 \exists y_2 (F y_1 y_2 \wedge \neg F y_1 y_2) \quad (11.1)$$

$$\exists y_1 \forall x_1 F x_1 x_1 y_1 \wedge \exists y_2 \forall x_2 \neg F x_2 y_2 x_2 \not\vdash \exists y_1 \exists y_2 (F y_2 y_2 y_1 \wedge \neg F y_2 y_2 y_1) \quad (11.2)$$

$$\exists y_1 \exists y_2 \forall x_1 \exists y_3 (F y_1 x_1 y_3 \wedge \neg F x_1 y_2 y_3) \not\vdash \exists y_1 \exists y_2 \exists y_3 (F y_1 y_2 y_3 \wedge \neg F y_1 y_2 y_3) \quad (11.3)$$

Formula (8.7) and the following paradigmatic examples of non-contradictory, anti-prenex, $\exists M$ -optimized formulae ψ'' cannot be transformed into explicit contradictions because of condition C2:

$$\forall x_1 \exists y_1 F x_1 y_1 \wedge \forall x_2 F x_2 x_2 \not\vdash \exists y_1 (F y_1 y_1 \wedge \neg F y_1 y_1) \quad (11.4)$$

$$\forall x_1 \exists y_1 (F x_1 y_1 \wedge \exists y_2 \neg F y_2 y_1) \not\vdash \exists y_1 \exists y_2 (F y_2 y_1 \wedge \neg F y_2 y_1) \quad (11.5)$$

$$\forall x_1 \exists y_1 F x_1 y_1 \wedge \forall x_2 \exists y_2 \neg F y_2 x_2 \not\vdash \exists y_1 \exists y_2 (F y_2 y_1 \wedge \neg F y_2 y_1) \quad (11.6)$$

In each of these cases, no optimal prenex normal form ψ^* can be generated from the anti-prenex, $\exists M$ -optimized formula that would permit the necessary substitutions.

12. OPTIMIZED SEARCH FOR PROOFS OF CONTRADICTION

The decision procedure for subformulae ψ can be used to conduct an optimized search for proofs of contradiction for arbitrary NNFs ϕ . We presume that ϕ is transformed into an anti-prenex FOLDNF ϕ' .⁸ The single maximally subscripted disjuncts D_i of ϕ' can be optimized by deleting literals in D_i that are not members of any unifiable pair of literals from D_i ; for details, cf. Algorithm 13.2.

To decide whether a pair of connected literals from D_i is unifiable and thus capable of contributing to a proof of contradiction, subformulae ψ are generated. Let D_i be an NNF that contains at least two connected literals. Subformulae ψ with exactly two connected literals, L1 and L2, can then be generated from D_i as follows:

Algorithm 12.1.

- (1) Delete all literals in ϕ except L1 and L2.
- (2) Delete all quantifiers binding variables that do not occur in L1 or L2.
- (3) Delete all occurrences of \wedge or \vee except the occurrence that connects L1 and L2 in the logical hierarchy of D_i .

⁸Further optimization is possible by transforming ϕ into a logically equivalent *minimized* FOLDNF; cf. [Lampert(2017a)].

(4) If L1 and L2 are connected by \vee , replace \vee with \wedge .

Step (4) is justified by the fact that the unification of pairs of literals in proofs of contradiction presumes that the literals are connected by conjunction. This does not imply that the literals are also connected by conjunction in D_i . It may well be that proofs of contradiction depend on the multiplication of universally quantified expressions by $\wedge I$ to replace the x variables of the multiplied universal quantifier with different y variables (including y_0). If the multiplied expression contains \vee , then literals that were connected by \vee before the application of $\wedge I$ are additionally connected by \wedge after the application of $\wedge I$. A proof of contradiction may then depend on the unification of those literals that are connected by \wedge subsequent to the application of $\wedge I$. Example 4.1, p. 7, illustrates this: Subsequent to the application of $\wedge I$, occurrences of $Fx_1x_2x_6$ and $\neg Fx_3x_1x_7$ are connected not only by \vee but also by \wedge . Consequently, the derivatives $Fx_{1_2}x_{2_2}x_{6_2}$ and $\neg Fx_{3_1}x_{1_1}x_{7_1}$ of $Fx_1x_2x_6$ and $\neg Fx_3x_1x_7$ are connected not by disjunction but by conjunction subsequent to maximal subscription, and this constitutes the proof of contradiction.

In general, a proof of contradiction for a formula ϕ in accordance with the $\wedge I$ -minimal proof strategy in the NNF-calculus is based on transforming ϕ into a sat-equivalent, optimized FOLDNF ϕ' and then specifying, for each disjunct D_i in ϕ' , (i) an anti-prenex, $\wedge I$ -optimized disjunct D_i^* ; (ii) a minimal set L of unifiable literals from D_i^* that are unified in the proof; (iii) a minimal substitution list σ that specifies how each x variable must be replaced with a y variable to unify L; and (iv) an optimal prenex that permits the substitutions necessary according to σ and that can be generated from D_i^* .

Deciding whether a pair of connected literals in D_i is unifiable is equivalent to deciding whether the corresponding subformula ψ is contradictory. Since the latter is decidable by means of Theorem 11.1, the identification of unifiable pairs of literals can be utilized to decide whether certain necessary conditions for a proof of the contradictoriness of D_i are satisfied.

Theorem 12.2. *An NNF ϕ is not contradictory if the DNF matrix of a D_i of the FOLDNF ϕ' that is sat-equivalent to ϕ contains a disjunct that does not contain any unifiable pair of literals.*

Proof. This theorem follows from the facts that ϕ is contradictory iff each disjunct D_i of ϕ' is contradictory and that any D_i is contradictory only if an explicit contradiction can be derived, which requires that each disjunct of the DNF matrix of D_i must contain at least one unifiable pair of literals. \square

Thus, a necessary condition for contradiction can be specified with respect to sat-equivalence transformations to identify formulae that are not contradictory.

Independent of the generation of FOLDNFs and DNF matrices, the following can be stated:

Theorem 12.3. *\wedge -NNFs are decidable.*

Proof. According to Theorem 1.5, a \wedge -NNF ϕ is contradictory iff at least one subformula ψ from ϕ is contradictory and its contradictoriness is decidable according to Theorem 11.1. \square

Example 12.4. The following formula is not contradictory:

$$\forall x_1 \exists y_1 Fx_1y_1 \wedge \forall x_2 \neg Fx_2x_2 \tag{12.1}$$

This follows from the fact that ψ''' contains only the following optimized prenex normal form:

$$\forall x_1 \exists y_1 \forall x_2 (F x_1 y_1 \wedge \neg F x_2 x_2) \quad (12.2)$$

x_1 must be replaced with y_1 for unification, but C2 of Theorem 11.1 is satisfied.

Example 12.5. This example demonstrates how it can be proven that a \wedge -NNF is contradictory by deciding upon subformulae ψ :

$$\forall x_1 \exists y_2 (\neg G y_2 y_2 \wedge \exists y_1 (\forall x_3 F y_1 x_1 x_3 \wedge \forall x_2 \neg F x_2 y_1 y_1) \wedge G x_1 y_2) \quad (12.3)$$

Formula (12.3) contains the following two pairs of connected literals: $\{G x_1 y_2, \neg G y_2 y_2\}$ and $\{F y_1 x_1 x_3, \neg F x_2 y_1 y_1\}$. $\forall x_1 \exists y_2 (\neg G y_2 y_2 \wedge G x_1 y_2)$ is the subformula ψ of the first pair of literals. In this case, $\psi = \psi''$, $\psi''' = \{\psi\}$, and $\sigma = \{\{x_1, y_2\}\}$. Hence, C1 is not satisfied. C2, however, is satisfied since ϕ''' does not contain any optimized prenex normal form with $\forall x_1$ in the scope of $\exists y_2$. Therefore, $\{G x_1 y_2, \neg G y_2 y_2\}$ is not unifiable. Since both literals do not occur in any other pair of literals from (12.3), these literals can be deleted in formula (12.3) for efficiency in the search for proofs of contradiction. The resulting formula is the subformula ψ of the second pair of literals and is identical to (8.1). The anti-prenex, \exists M-optimized formula ψ'' is (8.6). σ is unambiguous: $\{\{x_1, y_1\}, \{x_2, y_1\}, \{x_3, y_1\}\}$ ($= \sigma(8.6)$). Therefore, C1 is not satisfied. ψ''' contains only one optimized prenex normal form, which is optimal:

$$\exists y_{1_2} \forall x_1 \exists y_{1_1} \forall x_3 \forall x_2 (F y_{1_1} x_1 x_3 \wedge \neg F x_2 y_{1_2} y_{1_2}) \quad (8.4)$$

Therefore, C2 is also not satisfied. Applying \forall E in accordance with $\sigma(8.6)$ results in the following explicit contradiction:

$$\exists y_{1_1} \exists y_{1_2} (F y_{1_1} y_{1_2} y_{1_2} \wedge \neg F y_{1_1} y_{1_2} y_{1_2}) \quad (12.4)$$

Therefore, the pair of literals $\{F y_1 x_1 x_3, \neg F x_2 y_1 y_1\}$ in (12.3) is unifiable, and consequently, (12.3) is contradictory.

13. EFFICIENT IDENTIFICATION OF UNIFIABLE PAIRS OF LITERALS

The described algorithm for identifying unifiable pairs of literals is cumbersome due to the generation of ψ , ψ' , ψ'' and ψ''' . For the deletion of literals that are not members of any pair of unifiable literals in an NNF ϕ , it would be desirable to instead apply criteria that directly refer to ϕ , such as in the case of identifying pairs of connected literals. The intention is to specify necessary conditions for unifiable pairs of literals, although these conditions, when taken together, need not be sufficient.

The following 3 criteria are necessary conditions for unifiable pairs of literals that directly refer to the syntactic properties of a maximally subscripted NNF ϕ . They can be justified by the fact that the syntactic properties of ϕ to which they refer are sufficient for C1 or C2 to be satisfied even after \exists M-optimization and the generation of all optimized prenex normal forms of ψ''' .

C1U1: This criterion checks whether condition C1 (i.e., an ambiguous substitution list σ) is satisfied if the x variables in both of a pair of literals are subscripted differently by using a subscript of -1 in L1 and a subscript of -2 in L2. If the application of Algorithm 9.1 results in an ambiguous substitution list σ under this condition, then C1 will be satisfied even after \exists M-optimization.

Example 13.1. This criterion is satisfied if, e.g., an x variable occurs in two positions in one literal and those positions are occupied by different y variables in the other literal; cf. formula (11.1). The criterion is also satisfied if two x variables occur in the same position in both literals and both x variables must be replaced with different y variables due to their occurrences in other positions; cf. formula (11.2).

C2U1: This criterion checks whether (i) a universally quantified variable ν_1 occurs together with an existentially quantified variable μ in L1 (resp. L2), (ii) $\exists\mu$ is in the scope of $\forall\nu_1$ in ϕ (or ψ), (iii) ν_1 occupies the same position in L1 (resp. L2) as that of a universally quantified variable ν_2 in L2 (resp. L1), and (iv) ν_2 occupies the same position in L2 (resp. L1) as that of μ in L1 (resp. L2); cf. formula (11.4). In this case, C2 will be satisfied even after $\exists\text{M}$ -optimization and the generation of all optimized prenexes.

C3U1: This criterion checks whether (i) a universally quantified x variable ν_1 must be replaced with an existentially quantified variable μ_2 , (ii) a universally quantified x variable ν_2 must be replaced with an existentially quantified variable μ_1 , (iii) ν_1 and μ_1 occur in one literal while ν_2 and μ_2 occur in the other literal, and (iv) $\exists\mu_1$ is in the scope of $\forall\nu_1$ while $\exists\mu_2$ is in the scope of $\forall\nu_2$ in ϕ (or ψ); cf. formula (11.6). In this case, C2 will be satisfied even after $\exists\text{M}$ -optimization and the generation of all optimized prenexes.

Given a set of pairs of connected literals, C1U1-C3U1 enable one to identify a reduced set of pairs of connected literals that is guaranteed to contain all unifiable pairs of literals (= set 1). However, not all pairs of connected literals within this set are necessarily unifiable.

To identify all and only unifiable pairs of literals, one must generate the anti-prenex, $\exists\text{M}$ -optimized subformula ψ'' for each pair in set 1 and apply the following criteria:

C4U1: This criterion checks whether the substitution list σ of a pair of literals is ambiguous.

C5U1: This criterion checks whether σ contains an x list with an x variable ν and a y variable μ such that $\exists\mu$ is in the scope of $\forall\nu$ in ψ'' .

Given ψ'' , C4U1 decides whether C1 is satisfied, and C3U1 and C5U1 decide whether C2 is satisfied. The latter is true because the scopes cannot be minimized any further in ψ'' ; cf. Theorem 7.13. Given maximally minimized scopes, only two conditions prevent the generation of optimal prenexes: either $\exists\mu$ is in the scope of $\forall\nu$ in ψ'' (case 1), or ψ'' is of the form $\dots(\dots\forall\nu_1\dots\exists\mu_1\dots A(\mu_1, \nu_1) \wedge \dots\forall\nu_2\dots\exists\mu_2\dots B(\mu_2, \nu_2))$ (case 2). Case 1 ensures that $\exists\mu$ will also be in the scope of $\forall\nu$ in any optimized prenex normal form. C5U1 applies to this case. In case 2, any optimized prenex will contain either $\forall\nu_1$ to the left of $\exists\mu_2$ or $\forall\nu_2$ to the left of $\exists\mu_1$. Therefore, no optimal prenex exists such that $\exists\nu_1$ is to the left of $\forall\nu_2$ and $\exists\nu_2$ is to the left of $\forall\nu_1$. C3U1 applies to this case. In all other cases, however, either $\forall\nu$ is already in the scope of the corresponding existential quantifier $\exists\mu$ in ψ'' , if ν is to be replaced with μ for unification, or the strategy applied in Algorithm 10.1 of placing existential quantifiers to the left of universal quantifiers if possible will generate an optimal prenex. Therefore, it is sufficient to refer to ψ'' to decide the unifiability of the pairs in set 1.

C3U1 can already be applied to ϕ or ψ independent of the generation of ψ'' because $\exists\text{M}$ -optimization cannot cause C3U1 to not be satisfied in ψ'' when it is satisfied in ϕ or ψ .

Hence, it is sufficient to apply criteria C4U1 and C5U1 to the pairs of literals in set 1 to identify the unifiable pairs of literals.

Therefore, the following algorithm can be utilized for an optimized proof search by deleting literals in the NNF ϕ :

Algorithm 13.2.

- (1) Generate the set of all pairs of connected literals from ϕ .
- (2) Apply C1U1-C3U1 to this set to minimize it.
- (3) For each member of this set, generate the subformula ϕ'' and apply C4U1 and C5U1 to minimize it. The resulting set A contains all unifiable pairs of literals from ϕ .
- (4) Generate a set B consisting of all literals from ϕ that are not contained in set A.
- (5) Replace the literals from set B in ϕ with the expression 'sat'. The result is ϕ^* .
- (6) Delete all quantifiers in ϕ^* that do not bind a variable that occurs in some literal. The result is ϕ^{**} .
- (7) Iteratively apply the following rules from right to left, beginning with ϕ^{**} , as long as the resulting expression is either 'sat' or no longer contains 'sat':

$$\begin{array}{lcl} \text{sat} \wedge A & \dashv\vdash_{\text{sat}} & A \quad \text{SAT1} \\ \text{sat} \vee A & \dashv\vdash_{\text{sat}} & \text{sat} \quad \text{SAT2} \end{array}$$

Table 8: *sat* Rules

The result is ϕ^{***} : a formula that is sat-equivalent to ϕ and contains no literals that cannot contribute to any proof of contradiction for ϕ .

14. OUTLOOK

Given an anti-prenex FOLDNF with disjuncts D_i resulting from Algorithm 13.2 that (i) are not decidable with regard to their contradictoriness by means of Theorem 12.3 because they contain \vee and (ii) cannot be identified as non-contradictory by means of Theorem 12.2, the question arises of how the contradictoriness of such disjuncts D_i can be decided. According to the \wedge I-minimal proof strategy, this question boils down to the question of how to compute the minimal number of applications of \wedge I necessary to unify a minimal set of unifiable pairs of literals in general. Simple examples such as formula (4.1) show that applications of \wedge I are indispensable even if one allows for \exists M.⁹ However, if a disjunct D_i is contradictory, then a \wedge I-minimal proof of contradiction for D_i in the NNF-calculus is guaranteed to exist. Therefore, it is sufficient to restrict the search for proofs of contradiction to a systematic search for \wedge I-minimal proofs.

According to the \wedge I-minimal proof strategy, the necessary applications of \wedge I depend on the following parameters:

- (1) a minimal set L of pairs of unifiable literals that is sufficient for a proof of contradiction given the unification of its members,
- (2) a minimal list σ of substitutions that specifies how to replace x variables with y variables to unify the pairs in L, and
- (3) optimized prenexes \wp with respect to those quantifiers that bind variables of σ .

⁹Since applications of \wedge I are necessary for proofs of contradiction in general, it is convenient to dispense with \exists M altogether when one is considering general \wedge I-minimal proofs of D_i in accordance with the \wedge I-minimal proof strategy in the NNF-calculus.

All alternatives for unifying minimal sets of unifiable pairs of literals through the minimal number of substitutions in relation to all kinds of optimized prenexes must be considered in a systematic search for \wedge I-minimal proofs. As soon as a minimal set L with an unambiguous substitution list σ and an optimal prenex \wp is found, a \wedge I-minimal proof for D_i has been identified. However, if this does not occur, \wedge I must be applied to generate optimal prenexes by stipulating that x variables must be replaced with y_0 in addition to other substitutions or to separate x variables.

The question is when, in the search for \wedge I-minimal proofs, a search path can be abandoned because it is decidable based on purely syntactic criteria that no \wedge I-minimal proof can be found along that proof path. A prominent candidate for such a criterion that can be only vaguely specified at this point is as follows: On a certain proof path, as soon as the multiplication of a universally quantified expression through the application of \wedge I causes it to be necessary for the universally quantified expression resulting from this multiplication to, in turn, be multiplied by \wedge I to achieve similar substitutions of similar unifiable pairs of literals, it can be concluded that such repeated applications of \wedge I cannot be a necessary part of a \wedge I-minimal proof, and therefore, this proof path can be abandoned. Substitutions and pairs of literals are similar if they are identical in terms of their level-1 indices. Here, we presume that variables are maximally subscripted subsequent to the application of \wedge I by increasing the level of their indices by 1; cf. p. 7. The stated conditions can be identified syntactically if one keeps records of all applications of \wedge I in which universally quantified expressions are multiplied to unify certain unifiable pairs of literals by means of certain substitutions. I refer to this criterion for abandoning a proof path when applications of \wedge I on that proof path induce similar applications of \wedge I for similar substitutions in similar unifiable pairs of literals as the ‘loop criterion’.

For a \wedge I-minimal proof strategy, the relevant questions are how to restrict the search for proofs of contradiction to the search for \wedge I-minimal proofs and at what point the search for such proofs can finally be abandoned on all proof paths in the case of non-contradictory D_i due to a criterion such as the loop criterion. The specific algorithmic definition of a systematic search strategy for \wedge I-minimal proofs in fragments of FOL other than \wedge I-NNFs will require further detailed investigation and justification. The intention of this paper was to explain the underlying idea of such a proof search with respect to the most simple example, in which one can dispense with applications of \wedge I altogether. This might encourage, for more general fragments of FOL, a search for decision procedures that are analogous to the presented systematic search, which is based on nothing but purely proof-theoretic reasoning, for \wedge I-minimal proofs in the NNF-calculus.

REFERENCES

- [Boerger et al.] Börger, E., Grädel, E. & Gurevich, Y.: *The Classical Decision Problem*, Springer 2001.
- [Lampert(2017a)] Lampert, T. 2017: “Minimizing Disjunctive Normal Forms of Pure First-order Logic”, *Logic Journal of the IGPL* 25.3, 325-347.
- [Lampert(2017b)] Lampert, T. 2017: “Wittgenstein’s *ab*-notation: An Iconic Proof Procedure”, *History and Philosophy of Logic* 38.3, 239-262.
- [Nonnengart & Weidenbach] Nonnengart, A. & Weidenbach, C.: “Computing small clausal normal forms”, in: Robinson, A. & Voronkov, A. (eds.), *Handbook of Automated Reasoning I*, Elsevier, Amsterdam u.a., 2001, pp. 335-367.